VO 3
Datenfluss und andere Formen des Datenaustauschs

THEMEN

A. Asynchrone Kommunikation
B. Queues
C. Ereignisgesteuerte Programmierung
D. Gerätesteuerung

A. Asynchrone Kommunikation

LabVIEW ist eine datenflussorientierte Programmiersprache.

- Funktionen sind von Daten anderer Funktionen abhängig.
- Abhängige Funktionen werden erst ausgeführt, wenn zuvor abzuschließende Vorgänge fertig sind.
- Daten werden über Verbindungen weitergeleitet.

Zuweilen muss der Datenfluss jedoch unterbrochen und durch asynchrone Datenübertragung ersetzt werden.

Asynchrone Datenübertragung

Asynchrone Datenübertragung: Daten werden ohne Verbindungen übertragen

- Folgende Objekte tauschen Daten asynchron aus:
  - Parallele Schleifen
  - Programmoberfläche und Blockdiagramm
  - VIs
  - Applikationsinstanzen (LabVIEW-Projekte, EXE-Dateien etc.)
- Folgende Angaben werden ausgetauscht:
  - Daten
  - Benachrichtigungen

B. Queues

Queues

Queue-Funktionen

Erzeuger/Verbraucher-Entwurfsmuster (Daten)

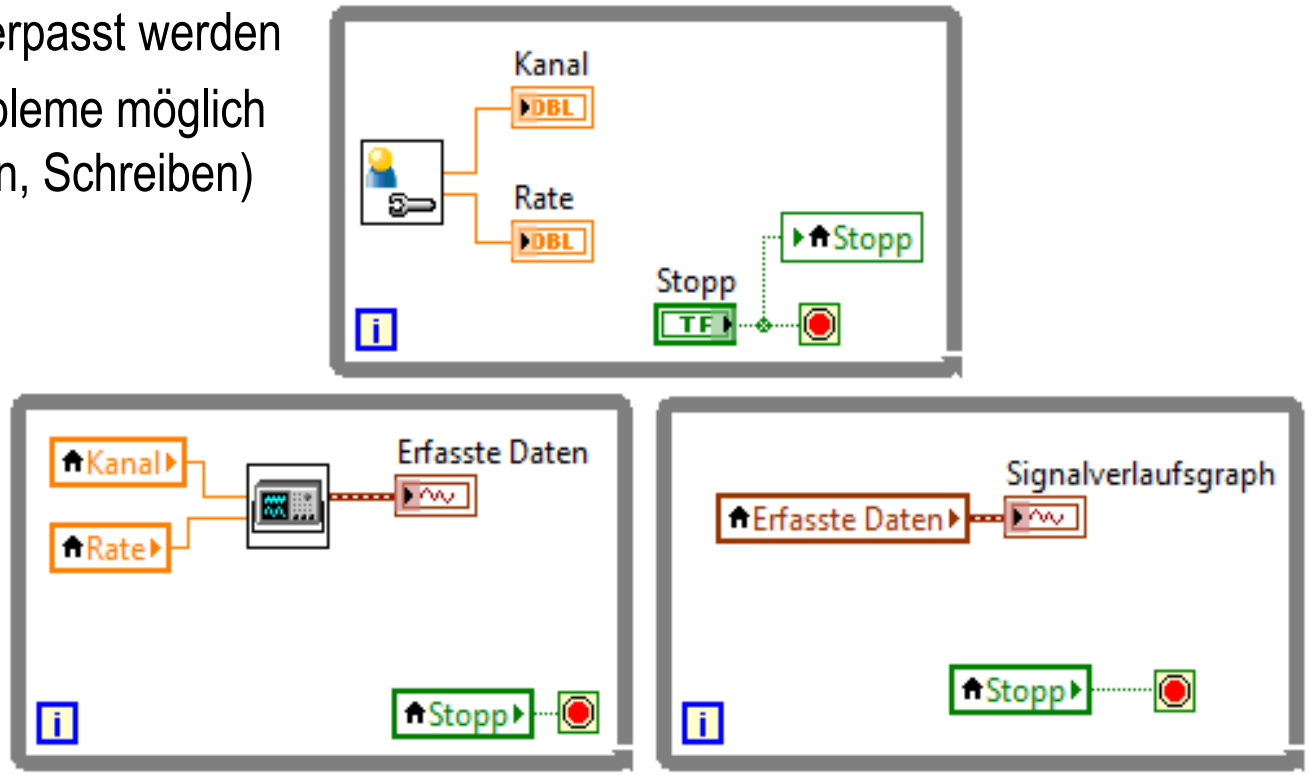
Queues

- Ermöglichen den Datenaustausch zwischen parallelen Schleifen
- Speichern mehrere Sätze von Daten (z. B. Pufferdaten)
- Arbeiten standardmäßig nach dem FIFO-Prinzip (First In, First Out)
- Können Daten jeden Typs enthalten

Nachteile von Variablen

Nachteile der Nutzung von Variablen für den Datenaustausch zwischen Schleifen:

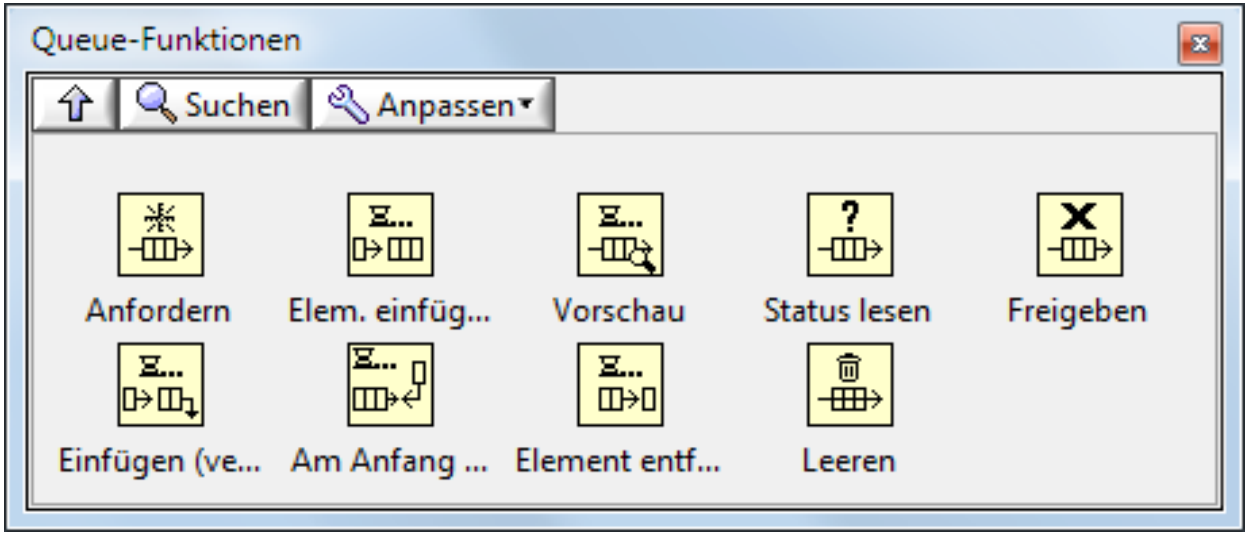
- Daten können doppelt gelesen werden
- Es können Daten verpasst werden
- Es sind Laufzeitprobleme möglich (Lesen, Modifizieren, Schreiben)



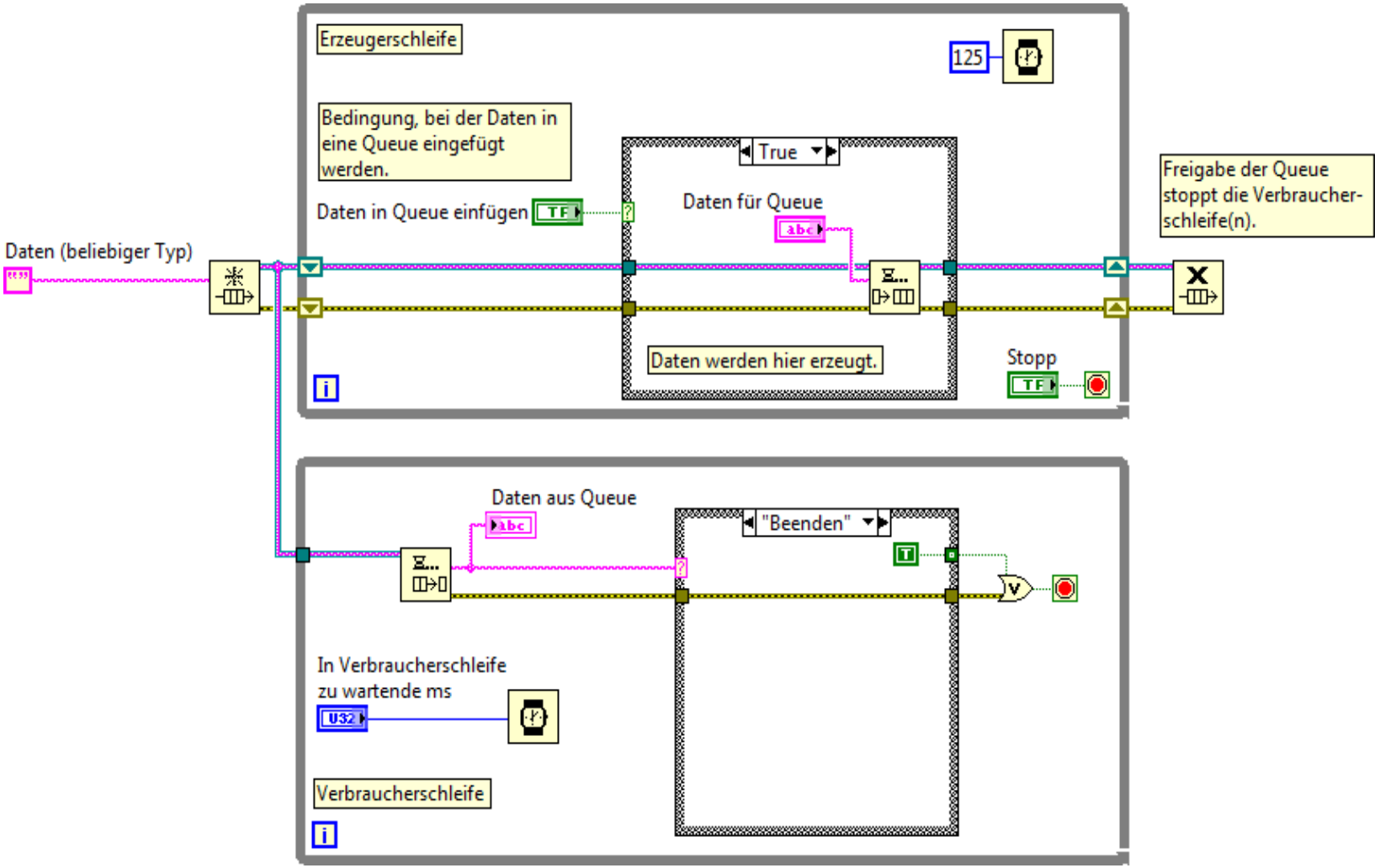
Queue-Funktionen

Die mit Queue-Funktionen erstellten Queues dienen dem Datenaustausch zwischen:

- Bestandteilen eines VIs
- Unterschiedlichen VIs



Erzeuger/Verbraucher-Entwurfsmuster (Daten)



C. Ereignisgesteuerte Programmierung

Ereignisse – Definition

Ereignisgesteuerte Programmierung – Definition

Vergleich von Polling und Ereignisstrukturen

Bestandteile einer Ereignisstruktur

Konfigurieren der Ereignisstruktur

Empfehlungen und Warnungen

Ereignisse

Ereignis – Asynchrone Meldung darüber, dass etwas stattgefunden hat

- Ereignisse werden durch den Benutzer, andere Bestandteile des VIs oder durch externe Signale ausgelöst
- Ereignisse sind Änderungen an Ereignisquellen
  - Beispiel: Wertänderung an Frontpanel-Elementen

Ereignisgesteuerte Programmierung

Ereignisgesteuerte Programmierung – Programmiermethode, bei der das Programm vor dem Ausführen von Funktionen auf das Eintreten eines Ereignisses wartet

Polling im Vergleich zu Ereignisstrukturen

Polling

- Methode der ereignisgesteuerten Programmierung, bei der eine kontinuierlich ausgeführte Schleife fortlaufend auf Änderungen prüft
- Frontpanel-Polling beansprucht viel Prozessorzeit
- Sehr schnelle Änderungen können beim Polling übersehen werden

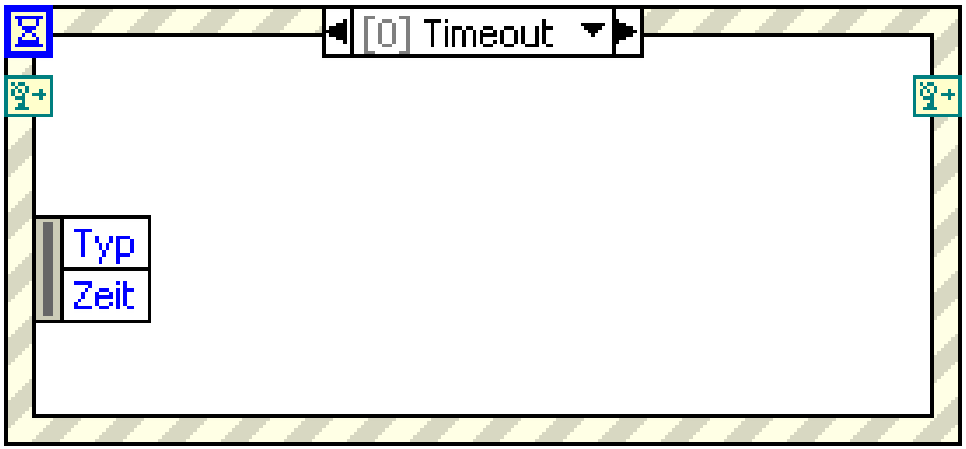
Ereignisstrukturen

- Bei der Nutzung von Ereignisstrukturen entfällt die Notwendigkeit des Frontpanel-Pollings
- Vorteile von Ereignisstrukturen:
  - Verringerte CPU-Last
  - Einfacheres Blockdiagramm
  - Sicherheit, dass das Blockdiagramm auf alle Handlungen des Benutzers reagiert

Verwenden von Ereignisstrukturen für die ereignisgesteuerte Programmierung

Ereignisstrukturen funktionieren wie Case-Strukturen, nur dass sie das VI "Auf Meldung warten" enthalten

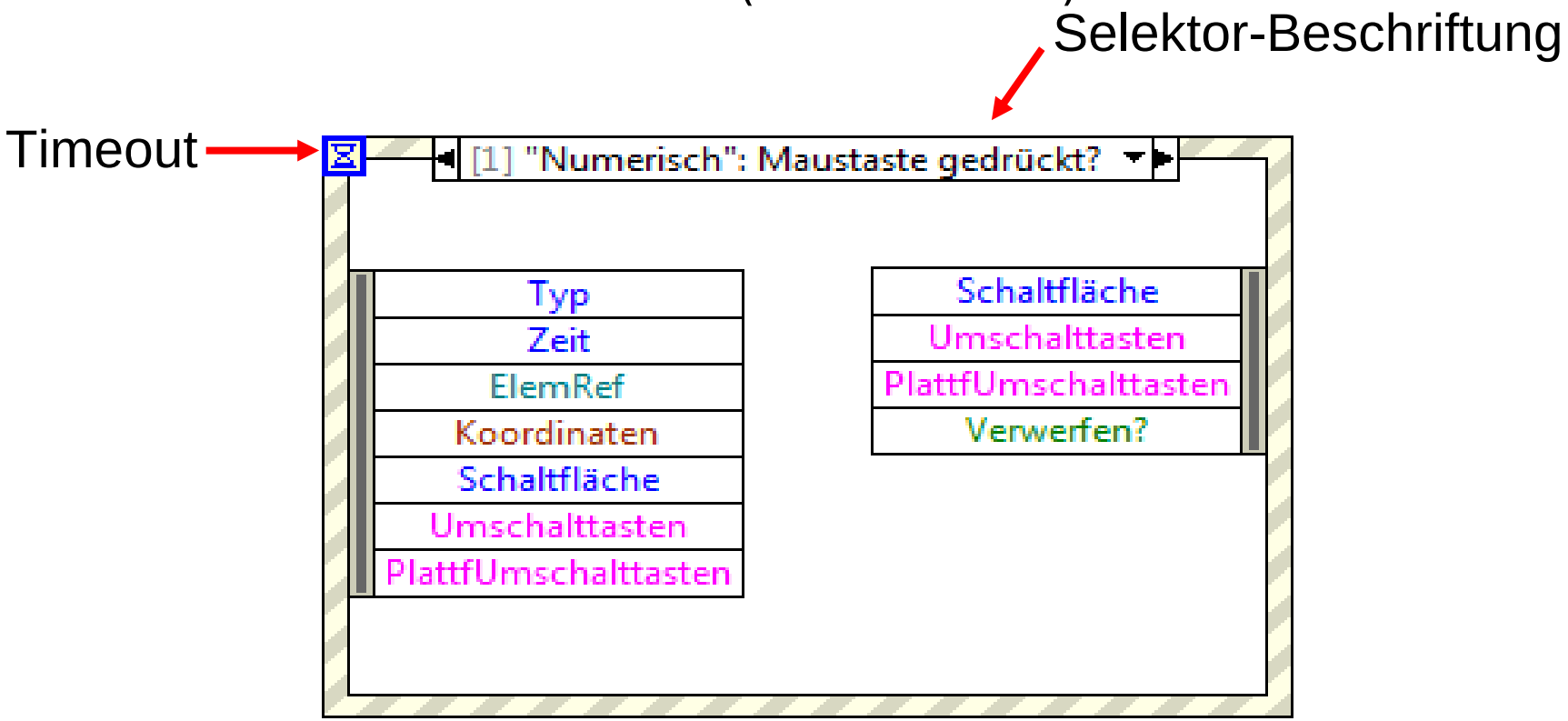
Ereignisstrukturen verarbeiten Ereignisse der Benutzeroberfläche (statische Ereignisse) folgender Art:



- Mausklick
- Tastendruck
- Wertänderung

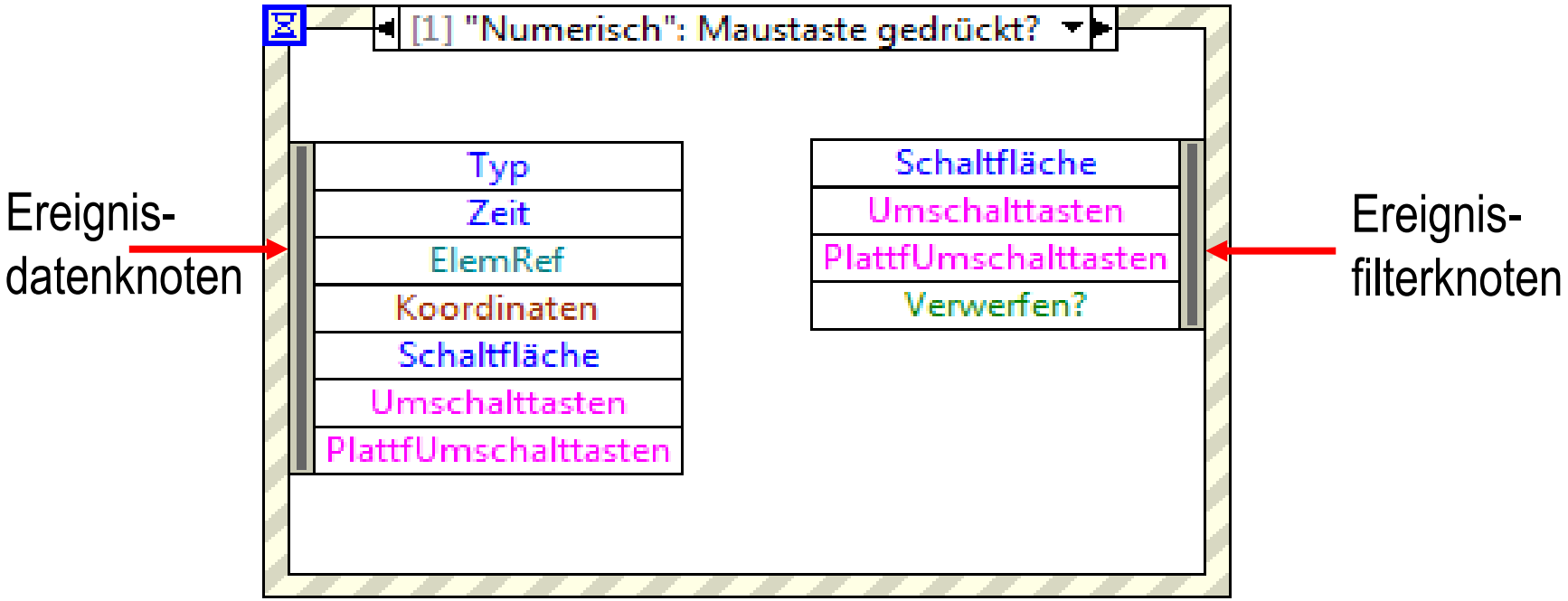
Bestandteile einer Ereignisstruktur

- Selektor-Beschriftung: Zeigt den Ereignis-Case an
- Timeout: Zeit in ms, die auf ein Ereignis gewartet wird; der Standardwert lautet –1 (kein Zeitlimit)



Bestandteile einer Ereignisstruktur (Fortsetzung)

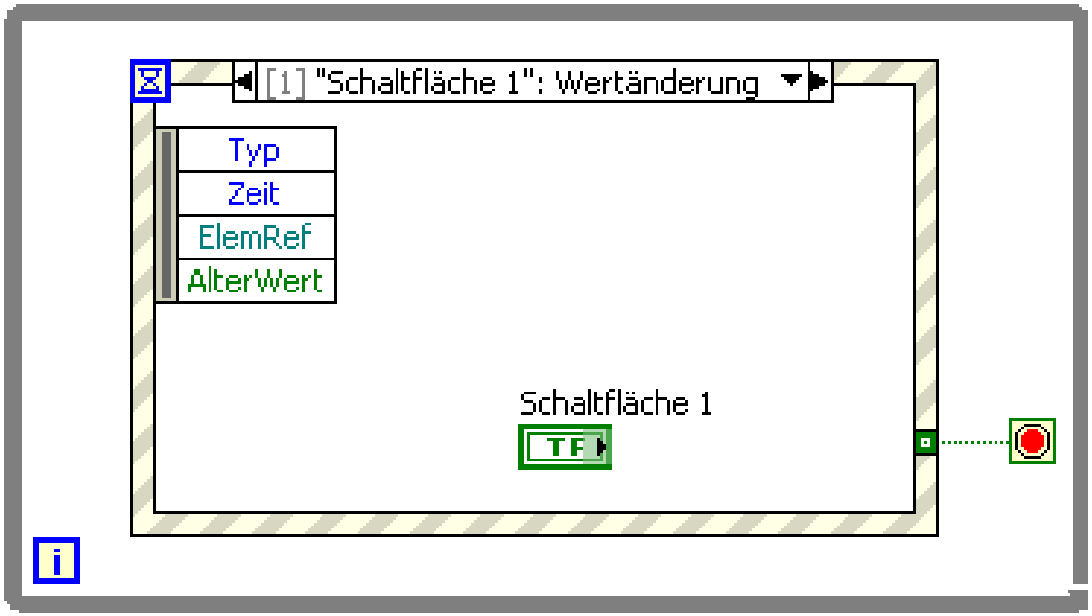
- Ereignisdatenknoten: Gibt ähnlich wie die Funktion "Nach Namen aufschlüsseln" die Parameter des Ereignisses aus
- Ereignisfilterknoten: Gibt Möglichkeiten zum benutzerdefinierten Verarbeiten des Ereignisses an



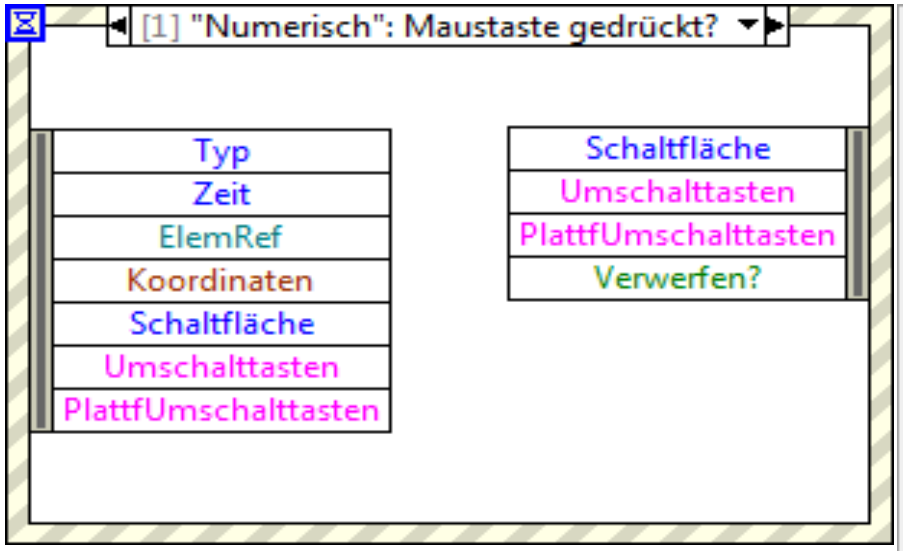
Verwenden von Ereignisstrukturen

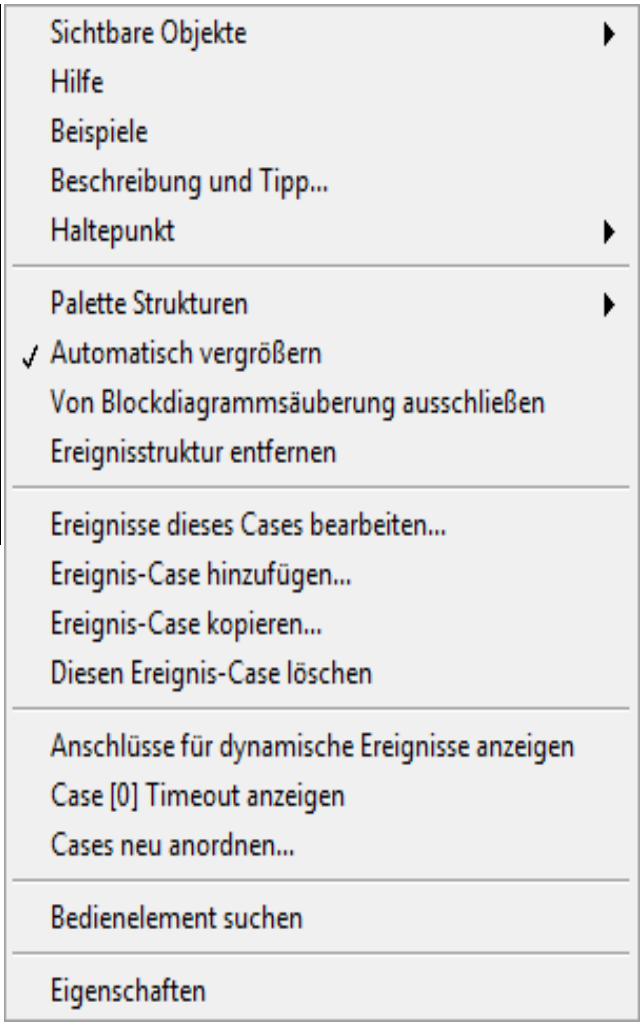
Ereignisstrukturen werden üblicherweise in While-Schleifen platziert

- Die Ereignisstruktur verarbeitet ein Ereignis pro Iteration der While-Schleife
- Bis zum Auftreten eines Ereignisses befindet sich die Ereignisstruktur im Wartemodus



Konfigurieren der Ereignisstruktur





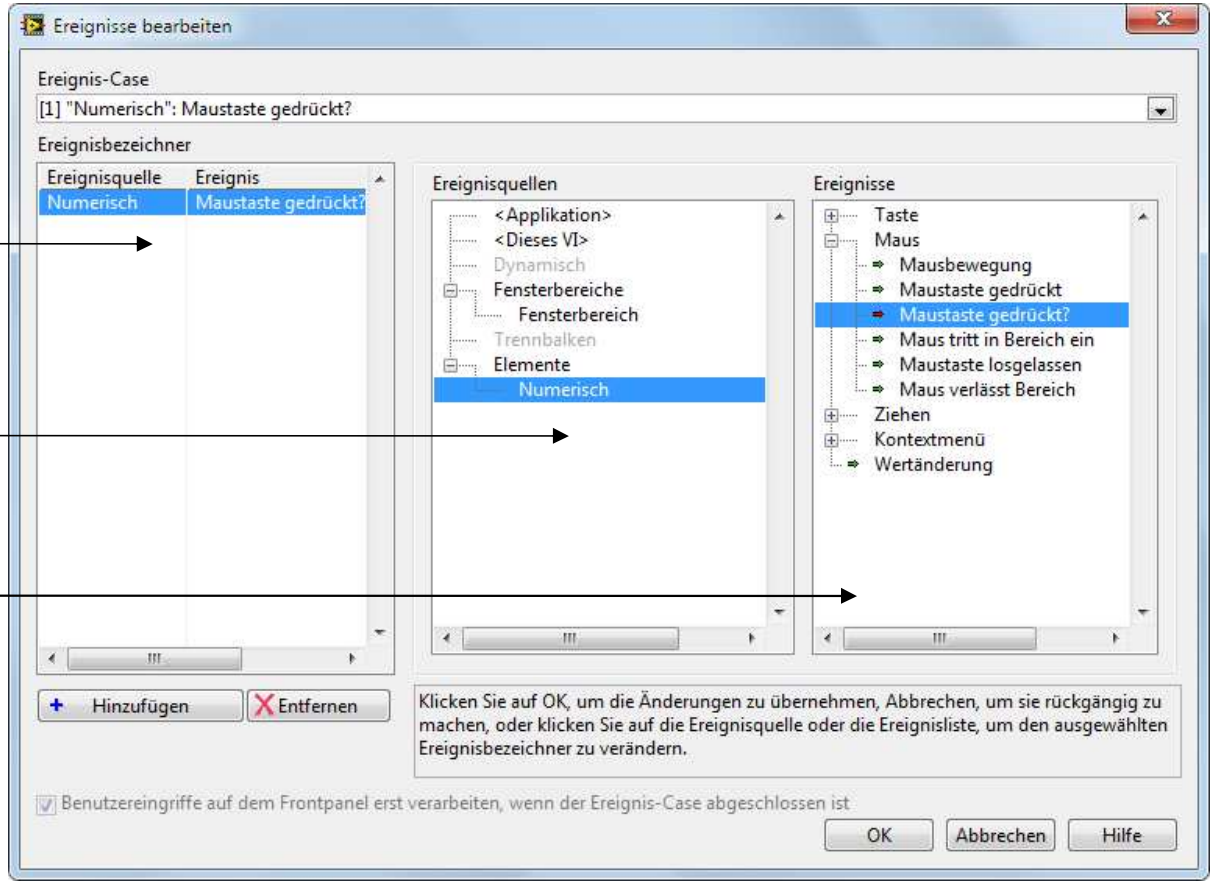
Zum Konfigurieren einer Ereignisstruktur klicken Sie den Rahmen der Struktur mit der rechten Maustaste an und wählen Sie **Ereignisse dieses Cases bearbeiten**

Dialogfeld "Ereignisse bearbeiten"

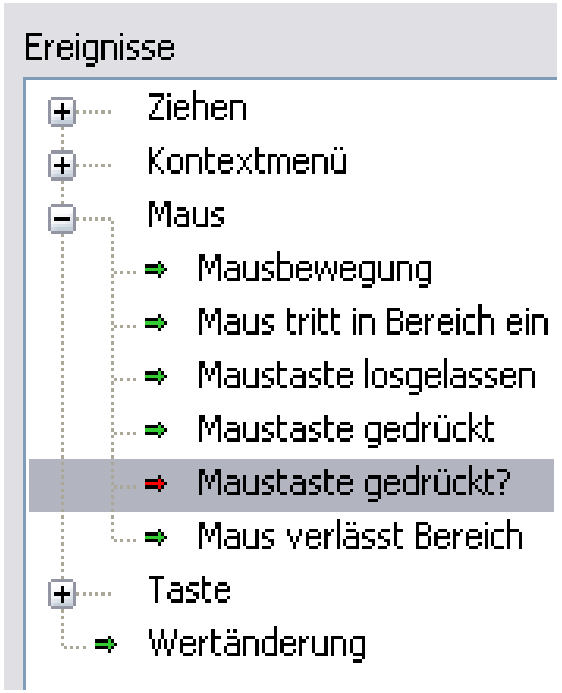
Konfigurierte Ereignisse

Ereignisquellen

Ereignisse



Melder- und Filterereignisse



Melderereignisse (grüner Pfeil)

Bedienhandlung bereits abgeschlossen und

Ereignis bereits von LabVIEW verarbeitet

Filterereignis (roter Pfeil)

Bedienhandlung bereits abgeschlossen und

Ereignis noch nicht von LabVIEW verarbeitet

Mit Filterereignissen ist die Standardreaktion auf Ereignisse veränderbar

Empfehlungen und Warnungen

- Ereignisstrukturen nicht außerhalb von Schleifen nutzen
- Nur eine Ereignisstruktur in eine Schleife einfügen
- Nicht zwei Ereignisstrukturen für dasselbe Ereignis konfigurieren
- Wertänderungen mit dem Ereignis "Wertänderung" erkennen
- Code zur Ereignisverarbeitung minimal halten
- Boolesche Elemente mit Latch-Funktion immer in Ereignis-Case einfügen, damit sie ordnungsgemäß funktionieren

D. Gerätesteuerung

Vorteile

GPIB

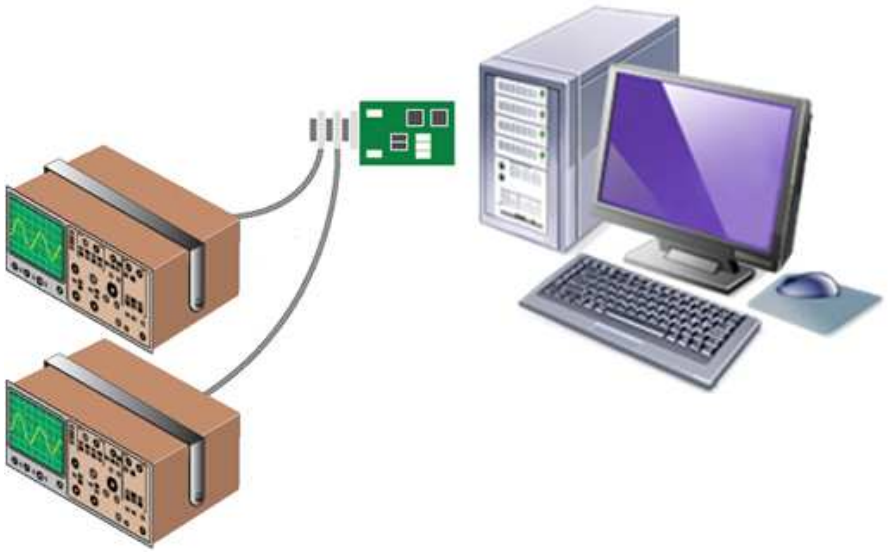
Gerätesteuerungssoftware

VISA

Gerätetreiber

Gerätesteuerung

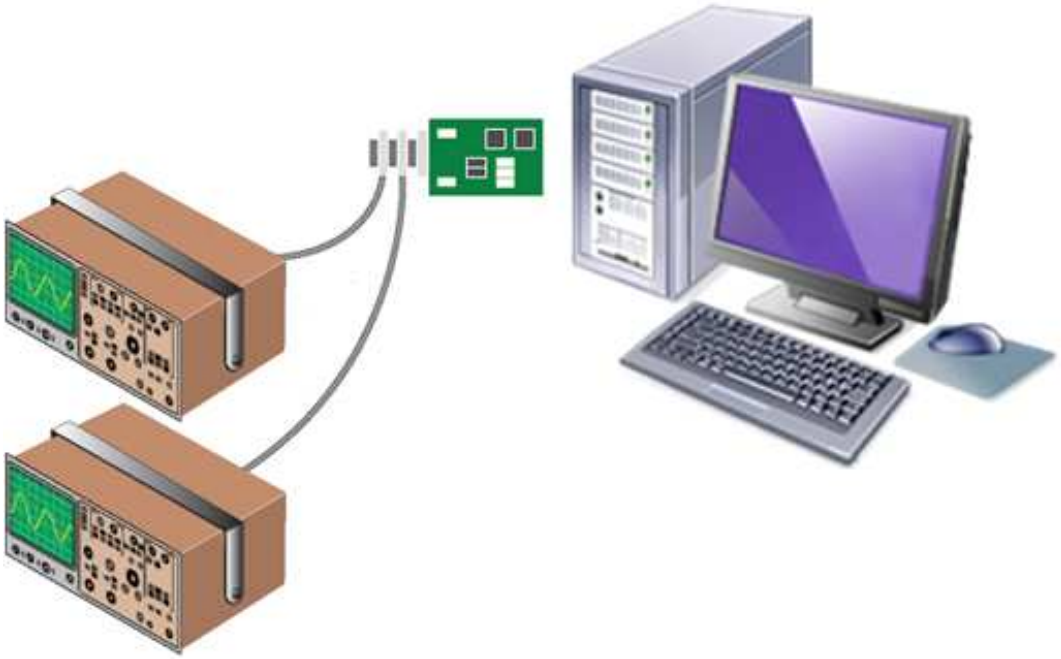
- Steuerung separater Geräte über einen speziellen Bus
- Geräte unterschiedlicher Kategorien kombinierbar
- Die wichtigsten Kenndaten des Geräts müssen bekannt sein, z. B. die verwendeten Kommunikationsprotokolle



Vorteile der Gerätesteuerung

- Prozessautomatisierung
- Zeitersparnis
- Eine Plattform für mehrere Tasks
- Einfach nutzbar
- Mit vielen Gerätetypen kompatibel

GPIB – General Purpose Interface Bus



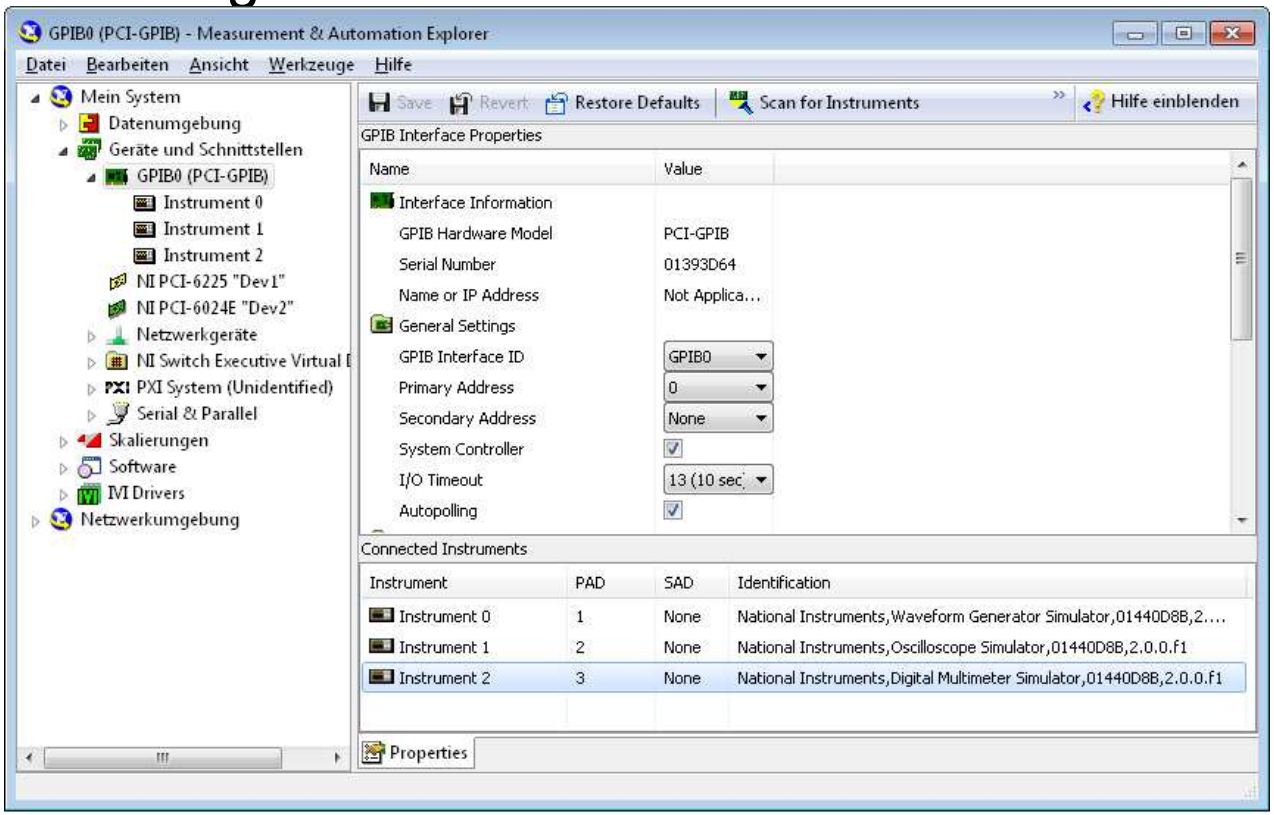
Standardschnittstelle für die Kommunikation zwischen einer Steuer-einheit (Controller) und Geräten unterschiedlicher Hersteller

GPIB

- Der Bus unterstützt einen System-Controller (in der Regel einen Computer) und bis zu 14 zusätzliche Geräte
- Der Controller:
  - Legt die Nachrichtenverbindungen fest
  - Sendet GPIB-Befehle
  - Reagiert auf Serviceanforderungen von Geräten
  - Übernimmt die Steuerung des Busses oder gibt sie ab

Verwendung von Gerätesteuerungssoftware

- Zu Schnittstellen wie GPIB gibt es eine Reihe von Treibern
- Hardwarekonfiguration im MAX



Übung 6-5
GPIB-Konfiguration im MAX

Konfiguration des NI-Gerätesimulators und Verwendung des MAX zum Untersuchen der Einstellungen der GPIB-Schnittstelle, Erkennen von Messgeräten und Kommunizieren mit einem Gerät

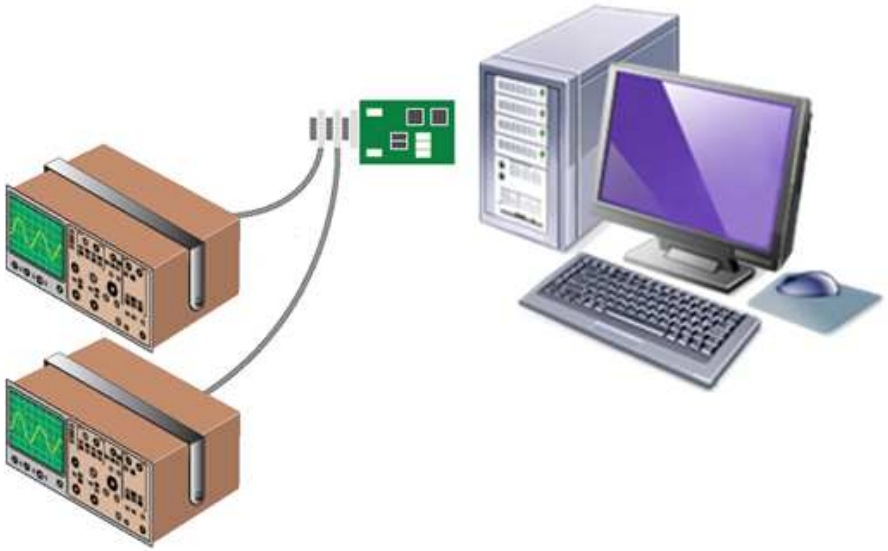
Übung 6-5
GPIB-Konfiguration im MAX

Überlegen Sie sich einen möglichen Einsatzzweck für eine Gerätesteuerungsanwendung

Programmierung der Gerätesteuerung

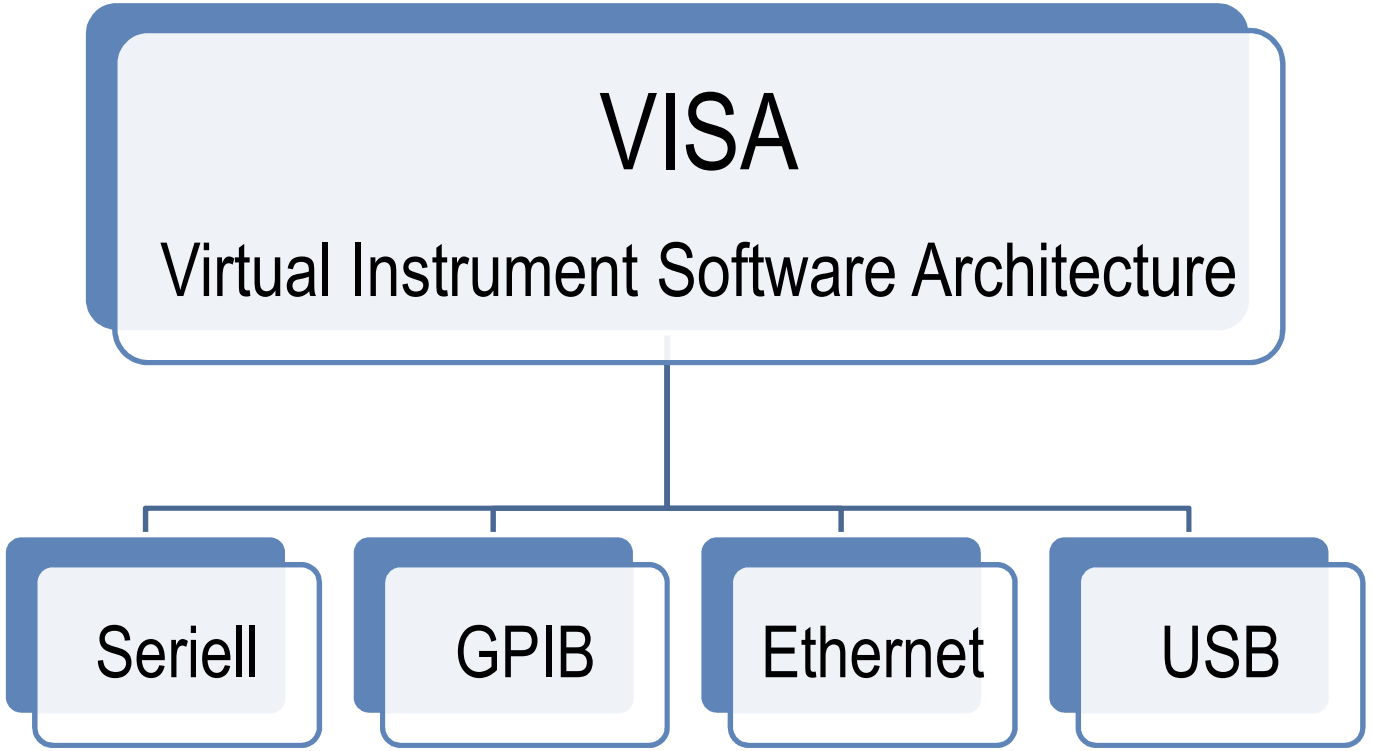
Virtual Instrument Software Architecture (VISA):

- Ist eine High-Level-API, die Low-Level-Treiber aufruft
- Steuert per GPIB, USB, Ethernet, serielles Kabel oder andere Schnittstelle angeschlossene Geräte und führt je nach Gerät die passenden Treiberaufrufe aus



VISA


VISA


VISA
Virtual Instrument Software Architecture
Seriell
GPIB
Ethernet
USB

VISA – Terminologie

- **Ressource**

  Elektrische Einheit im System einschl. der seriellen/parallelen Schnittstellen

- **Session**

  – Für jede Ressource einzeln zu erstellen (ähnlich wie bei einem Kommunikationskanal)

  – VISA-Session-Nummer als eindeutige Kennung (Referenz) des Geräts

  – Ist bei allen folgenden VISA-Operationen zu verwenden

- **Gerätedeskriptor**

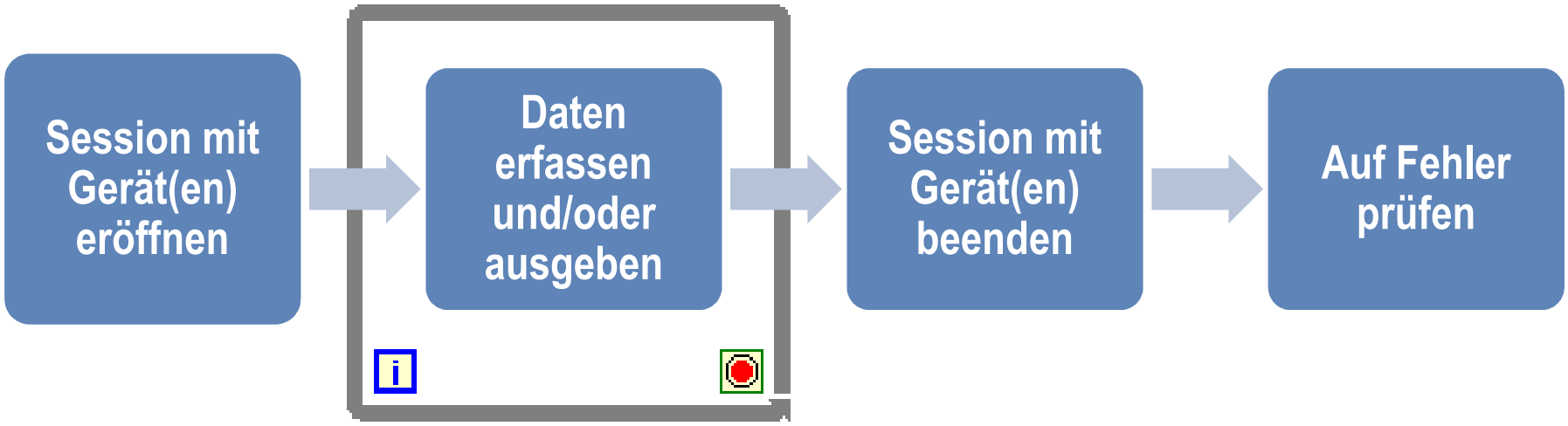
  Gibt die Art der Schnittstelle (GPIB, USB, TCP/IP, ASRL), die Adresse des Geräts und die Art der VISA-Session (INSTR oder Ereignis) an

VISA-Alias

Vom Benutzer festgelegter Name für ein Gerät oder eine Ressource, der statt des Gerätedeskriptors verwendet wird

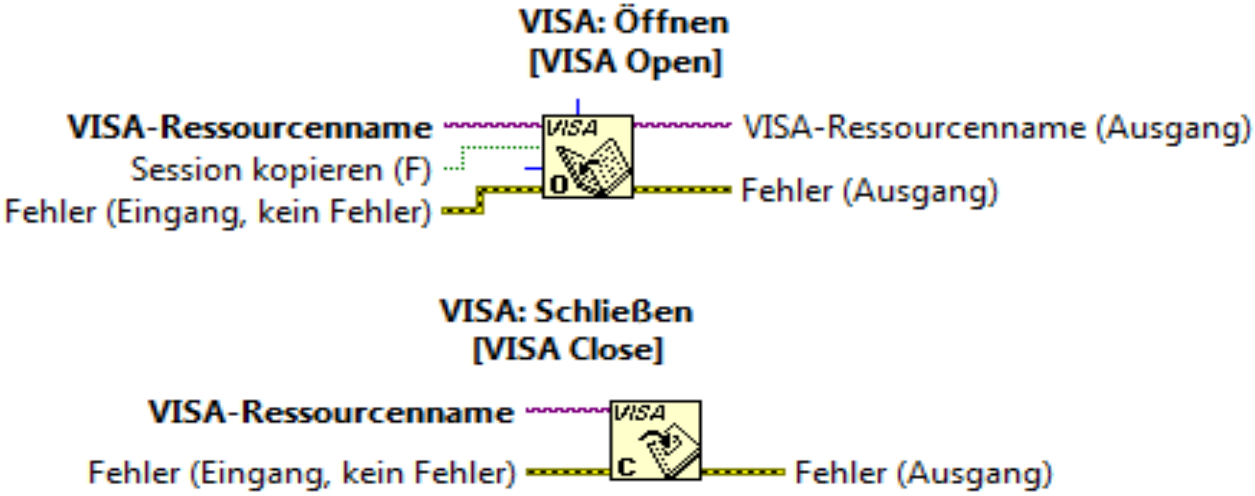
VISA-Programmierung

VISA-Funktionen arbeiten ähnlich wie Datei-I/O-Funktionen



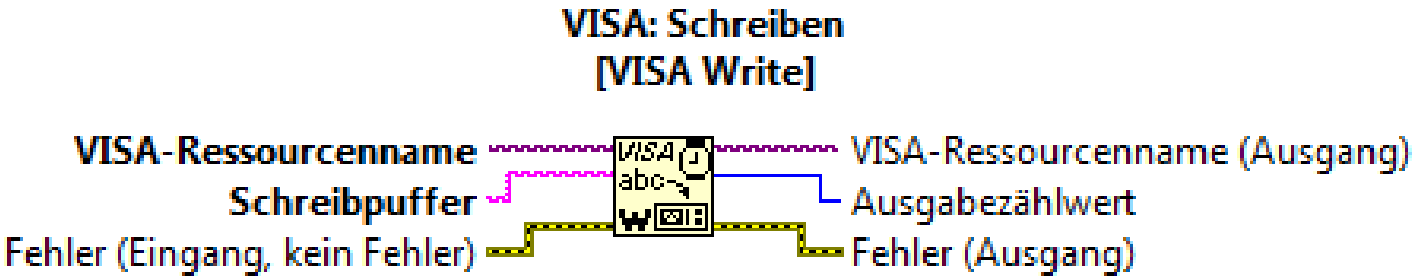
Funktionen "VISA: Öffnen" und "VISA: Schließen"

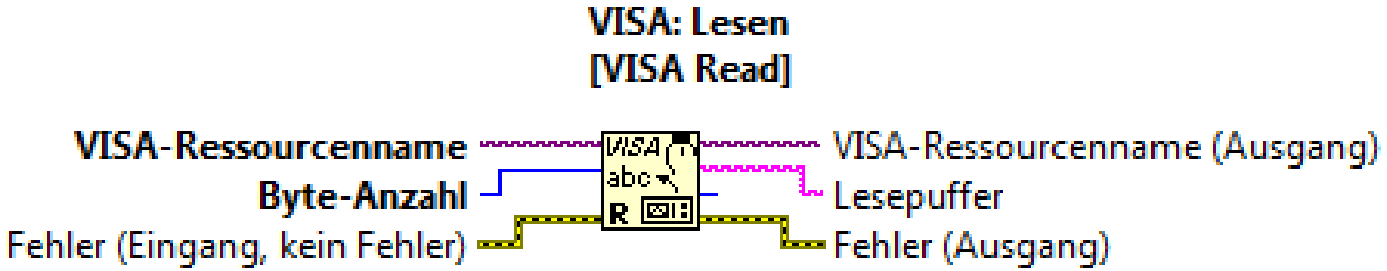
- Funktion "VISA: Öffnen"
  - Stellt eine Kommunikationsverbindung mit der Ressource her
  - Wird üblicherweise einmal pro Ressource genutzt
  - Gibt den Namen der VISA-Ressource aus



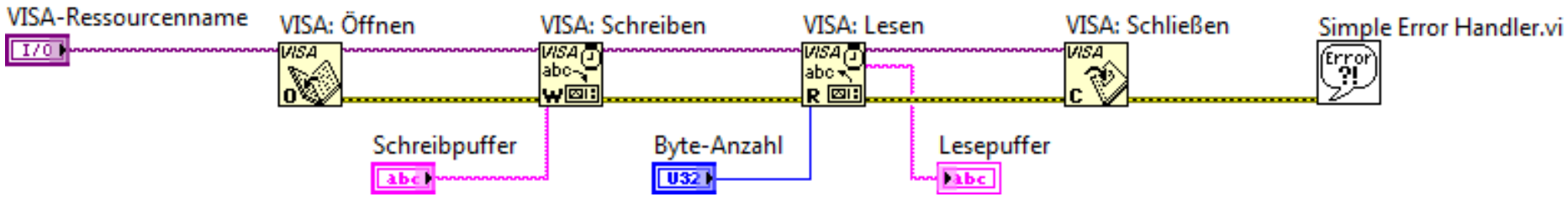
VISA-I/O-Funktionen

"VISA: Schreiben" und "VISA: Lesen"



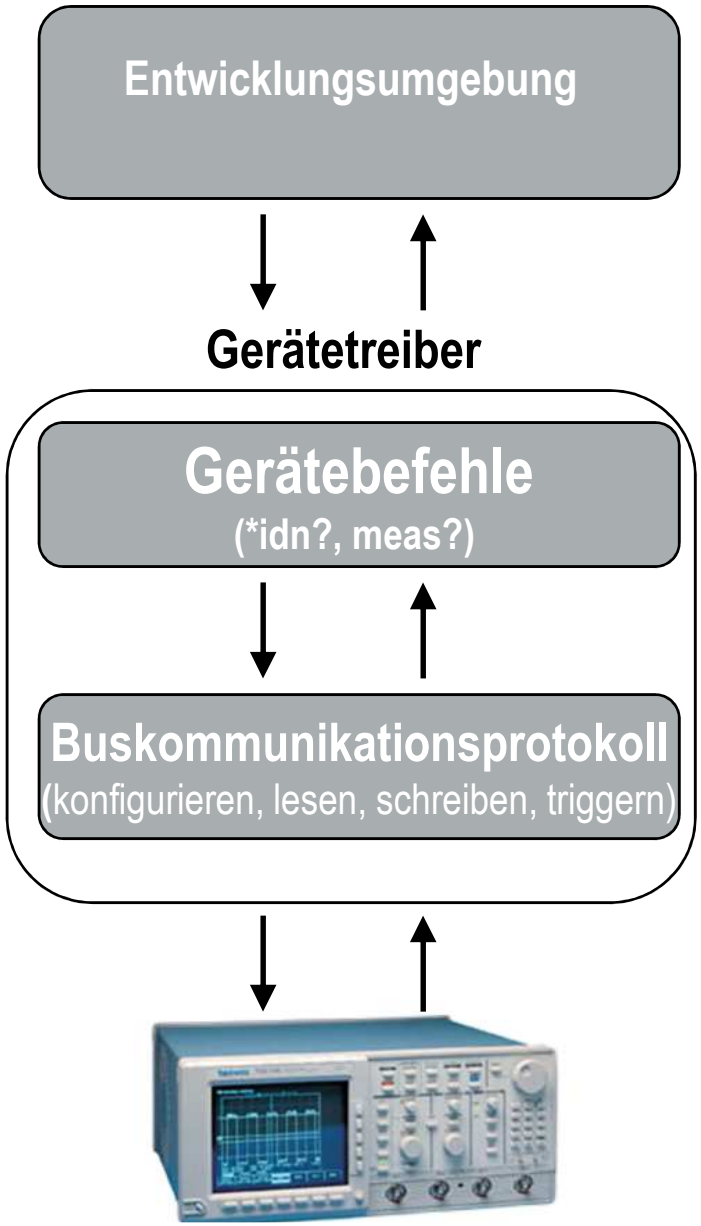


Beispiel zum Datenaustausch mit VISA



Gerätetreiber

- VIs zur Steuerung programmierbarer Geräte
  - Mehrere Arbeitsschritte pro VI
  - Nach Arbeitsschritten eingeteilt (z. B. Konfiguration oder Erfassung)
- Geringer Entwicklungsaufwand
  - Vereinfachte Steuerung
  - Wiederverwendbar
  - Gängige Architektur und Schnittstelle



Verwenden von Gerätetreibern

- Wenn Sie mit einem Treiber arbeiten, enthält dieser alle gerätespezifischen Befehle
- Bei einem Gerätewechsel müssen Sie die Treiber-VIs nur durch VIs für das neue Gerät ersetzen

Gerätetreiber – Suche nach Treibern

- Die meisten LabVIEW-Treiber für Plug-and-Play-Geräte finden Sie über die Suchmaschine für NI-Gerätetreiber
  - In LabVIEW: **Werkzeuge»Instrumentierung»Gerätetreiber suchen** oder **Hilfe»Gerätetreiber** suchen
  - Sucht auf ni.com nach Treibern
  - Auf ni.com/idnet können Sie auch selbst nach Treibern suchen
- Wenn Sie einen Gerätetreiber installieren:
  - Wird ein entsprechendes Beispielprogramm zur NI-Suchmaschine für Beispiele hinzugefügt
  - Werden Gerätetreiber-VIs zur Palette **Instrumenten-I/O»Gerätetreiber** der **Funktionen**-Palette hinzugefügt
  - Extrahiert die Gerätetreiber-VIs und die dazugehörigen Dateien in das Verzeichnis <LabVIEW>\instr.lib

Zusammenfassung – Quiz

1. Mit welchem der folgenden Elemente können Daten gepuffert werden?
   a) Queues
   b) Ereignisse
   c) Lokale Variablen

Zusammenfassung – Antworten

1. Mit welchem der folgenden Elemente können Daten gepuffert werden?
   a) **Queues**
   b) **Ereignisse**
   c) Lokale Variablen

Zusammenfassung – Bilden Sie Paare:

1. Queue anfordern 
2. Queue-Status lesen 
3. Queue freigeben 
4. Element einfügen 

a. Schließt Referenz auf Queue

b. Weist den Datentyp der Queue zu

c. Fügt an das Ende der Queue ein Element an

d. Legt die Anzahl der Elemente fest, die sich in der Queue befinden

Bilden Sie Paare (Antworten)

1. Queue anfordern 
2. Queue-Status lesen 
3. Queue freigeben 
4. Element einfügen 

a. Schließt Referenz auf Queue

b. Weist den Datentyp der Queue zu

c. Fügt an das Ende der Queue ein Element an

d. Legt die Anzahl der Elemente fest, die sich in der Queue befinden

Zusammenfassung – Quiz

3. Welche der folgenden Datentypen können in Queues verwendet werden?
   a) String
   b) Numerisch
   c) Enum
   d) Boolesches Array
   e) Cluster aus String und einem numerischen Element

Zusammenfassung – Antworten

3. Welche der folgenden Datentypen können in Queues verwendet werden?
   a) **String**
   b) **Numerisch**
   c) **Enum**
   d) **Boolesches Array**
   e) **Cluster aus String und einem numerischen Element**

Zusammenfassung – Quiz

4. Bei jeder Ausführung einer Ereignisstruktur kann immer nur ein Ereignis bearbeitet werden.
   a) Richtig
   b) Falsch

Zusammenfassung – Antworten

4. Bei jeder Ausführung einer Ereignisstruktur kann immer nur ein Ereignis bearbeitet werden.
   a) **Richtig**
   b) Falsch